

Algoritmos y Estructuras de Datos II

Primer parcial – Viernes 29 de abril de 2022

120

- Es posible tener una hoja (2 cartillas), escrita a mano, con los apuntes que se deseen.
- Cada ejercicio debe entregarse en hojas separadas.
- Incluir en cada hoja el número de orden asignado, número de libreta, número de hoja, apellido y nombre.
- Cada ejercicio se calificará con Perfecto, Aprobado, Regular, o Insuficiente.
- El parcial estará aprobado si las notas de los tres ejercicios son mejores o iguales a alguna de las siguientes combinaciones: {R, A, R}, {R, R, A} o {I, A, A}.
- Los ejercicios no se recuperan por separado.

1	2	3	A
P	A	A+	Alc.

Ej. 1.

Axiomatizar la función **tieneValor** que, dado un diccionario a naturales y un natural, establece si el diccionario tiene alguna clave cuyo significado sea el segundo parámetro. Esto es:

$$\text{tieneValor} : \text{dicc}(\alpha, \text{nat}) \times \text{nat} \rightarrow \text{bool}$$

Que cumple que:

$$(\forall a : \text{nat}, d : \text{dicc}(\alpha, \text{nat})) \text{ tieneValor}(d, a) \Leftrightarrow (\exists e : \alpha) e \in \text{claves}(d) \wedge \text{obtener}(d, e) = a$$

Ej. 2.

Un Videoclub es un negocio donde se tienen copias físicas de películas que pueden ser alquiladas. Los clientes alquilan las películas, se las llevan a su casa y luego las devuelven. Para cada película, existe una cantidad de días máxima en que esta puede ser tenida por una persona antes de devolverla. Si a un cliente se le vence el plazo para devolverla – se queda sin días para devolverla – automáticamente es desafiliado del videoclub. Cuando esto sucede, el videoclub va a su casa para notificarlo y recuperar las películas que estaban alquiladas por él (ahora ex) cliente. Además, como el videoclub está recién empezando, solo posee una única copia de cada película en su catálogo.

Se pide completar el TAD presentado a continuación agregando los axiomas para los observadores **díasRestantesAlquiler** y **clientes**.

Se dispone de la función auxiliar **restarUno**: $\text{dicc}(\alpha, \text{nat}) \rightarrow \text{dicc}(\alpha, \text{nat})$ que resta uno a cada valor definido en el diccionario (y que se define si algún significado es igual a 0) y la función auxiliar **tieneValor**: $\text{dicc}(\alpha, \text{nat}) \times \text{nat} \rightarrow \text{bool}$ presentada en el ejercicio 1 (que pueden suponer definida aunque no lo hayan hecho).

TAD CLIENTE es string
TAD PELICULA es string

TAD VIDEOCLUB

igualdad observacional
(=)

géneros vc

observadores básicos

clientes	: vc	$\rightarrow \text{conj}(\text{cliente})$	
pelis	: vc	$\rightarrow \text{dicc}(\text{pelicula}, \text{nat})$	
díasRestantesAlquiler	: vc \times cliente c	$\rightarrow \text{dicc}(\text{pelicula}, \text{nat})$	$\{c \in \text{clientes}(vc)\}$
díaActual	: vc	$\rightarrow \text{nat}$	

generadores

inaugurarVC	: $\text{conj}(\text{cliente}) \times \text{dicc}(\text{pelicula}, \text{nat}) \times ps$	$\rightarrow vc$	$\{\neg(\exists p \in \text{claves}(ps) \wedge \text{obtener}(ps, p) = 0)\}$
alquilar	: vc \times cliente $c \times$ pelicula p	$\rightarrow vc$	$\{c \in \text{clientes}(vc) \wedge p \in \text{claves}(\text{peliculas}(vc)) \wedge \neg(\exists c' \in \text{clientes}(vc) \wedge \text{def}^*(p, \text{díasRestantesAlquiler}(vc, c')))\}$

devolver	: $vc \times cliente \times pelicula \rightarrow vc$	$\{c \in clientes(v) \wedge def?(p, diasRestantesAlquiler(v, c))\}$
pasarDia	: $vc \rightarrow vc$	

axiomas (extracto)

$\forall v: videoclub, \forall c, c2: cliente, \forall p: pelicula, \forall ps: conj(pelicula), \forall cs: conj(cliente)$

diaActual(inaugurarVC(cs, ps))	$\equiv 0$
diaActual(alquilar(v, c, p))	$\equiv diaActual(v)$
diaActual(devolver(v, c, p))	$\equiv diaActual(v)$
diaActual(pasarDia(v))	$\equiv diaActual(v) + 1$
diasRestantesAlquiler(inaugurarVC(cs, ps), c2)	$\equiv \dots *$
diasRestantesAlquiler(alquilar(v, c, p), c2)	$\equiv \dots *$
diasRestantesAlquiler(devolver(v, c, p), c2)	$\equiv \dots *$
diasRestantesAlquiler(pasarDia(v), c2)	$\equiv \dots *$
clientes(inaugurarVC(cs, ps))	$\equiv \dots *$
clientes(alquilar(v, c, p))	$\equiv \dots *$
clientes(devolver(v, c, p))	$\equiv \dots *$
clientes(pasarDia(v))	$\equiv \dots$

Fin TAD

Como se puede ver en el TAD, el Videoclub cuenta con las siguientes funciones:

- **clientes** devuelve los clientes actualmente afiliados al videoclub.
- **pelis** devuelve un diccionario con la cantidad de días máximos que una película puede ser alquilada.
- **diasRestantesAlquiler** devuelve la cantidad de días restantes para cada copia retirada por un cliente.
- **díaActual** devuelve cuantos días pasaron desde la apertura del videoclub.
- **inaugurarVC** crea un videoclub con un conjunto de clientes y un diccionario que indica la cantidad máxima de días de alquiler para cada película.
- **alquilar** permite a un afiliado alquilar una película
- **devolver** permite a un afiliado devolver una película

Ej. 3.

La famosa emisora televisiva HBOBo está lanzando su nuevo reality show, Juego de Calamares, que sigue en vivo y en directo la evolución de una colonia de calamares muy competitivos. El tanque será aislado al comenzar la serie, es decir, no ingresarán nuevos calamares durante el juego. Los calamares se organizan siguiendo a otros calamares que consideran sus líderes. Cada calamar puede tener (o no) varios seguidores, mientras que un calamar puede seguir a lo sumo a un único otro calamar. En cualquier momento, un calamar puede cambiar a qué calamar sigue. Durante las luchas de poder, es común que un calamar avergüenze a otro (le pinta la cara). En estos casos, todos los seguidores del calamar avergonzado pasan a seguir al avergonzador. El calamar avergonzado no puede con su orgullo y se retira del juego. HBOBo nos encarga especificar usando TADs el comportamiento de los clamares durante el Juego de Calamares. Se desea saber en todo momento quienes van ganando el Juego de Calamares; a tal efecto se considera que un calamar va ganando cuando sus seguidores (la cantidad total) son los más numerosos.

Dada la definición del problema presentado, definir un TAD JUEGODECALAMARES a partir de los siguientes pasos:

- (a) Listar en lenguaje natural qué aspectos son relevantes para diferenciar dos Juegos de Calamares.
- (b) Expresar esto en un conjunto de observadores y una definición de igualdad observacional. Comentar qué parte del enunciado expresa cada observador.
- (c) Definir un conjunto de generadores que permitan generar todos los valores relevantes del TAD. Comentar qué parte del enunciado expresa cada generador.
- (d) Axiomatizar el/los observadore/s que indican las relaciones de seguimiento entre los calamares.

Ejercicio 1

$$\text{tieneValor}(d, n) \equiv \text{tieneValorAux}(d, \text{claves}(d), n)$$

$\text{tieneValorAux} : \text{dicc}(\alpha, \text{nat}) \times \text{conj}(\alpha) \times \text{claves} \times \text{nat} \times n \rightarrow \text{bool}$ ~~...~~
 $\{ \text{claves} \subseteq \text{claves}(d) \}$ ✓

$\text{tieneValorAux}(d, \text{claves}, n) \equiv \text{If } (\text{vacío?}(\text{claves}))$

Then: false
+ else: If (obtener(d, dameUno(claves)) = n)
fi
+ then: True
+ else: tieneValorAux(d, sinUno(claves), n)
fi ✓ OK!

Para la resolución se implementa una función auxiliar que recibe 2 datos un parámetro que debe ser ~~...~~ un subconjunto de las claves del diccionario. Se recibe: $\{ \text{claves} \subseteq \text{claves}(d) \}$

La idea de esta función es poder controlar con ayuda de las funciones "dameUno" x "sinUno" si el valor que se obtiene en la función obtener es igual al solicitado.

En caso de no encontrarse (cuando se terminen las claves por probar) se indica false. De encontrar ~~...~~ coincidencias, true. ✓

Ejercicio 2

Por qué?

// no tiene sentido decirle a cada cliente preguntar los restos. Si realmente no es problema de lo se retorna un diccionario vacío.

$\text{disponiblesAlquiler}(\text{Inaugurar}(c, p), v, c_2) \equiv \text{disponiblesAlquiler}(v, c_2)$ ← esto no es un dicc. vacío.

$\text{disponiblesAlquiler}(\text{Alquiler}(v, c, p), c_2) \equiv$

if ($c = c_2$)

+ then: definir($\text{disponiblesAlquiler}(v, c)$, p, devolver($\text{restos}(v)$, p))

+ else: $\text{disponiblesAlquiler}(v, c_2)$

fi ✓

// Al borrar la película del diccionario habilito a otro cliente a poder alquilarle, por lo que las restricciones de Alquiler(...)

$\text{disponiblesAlquiler}(\text{devolver}(v, c, p), c_2) \equiv$

if ($c = c_2$)

+ then: borrar($\text{disponiblesAlquiler}(v, c)$, p)

+ else: $\text{disponiblesAlquiler}(v, c_2)$

fi ✓

$\text{disponiblesAlquiler}(\text{Reservar}(v), c_2) \equiv \text{restarUno}(\text{disponiblesAlquiler}(v, c_2))$ ✓

// La función restarUno(...) retorna un dicc vacío si no tiene películas alquiladas al Cliente. Además de la naturaleza del tad y las abstracciones de indeterminación de restar uno se le debe ~~contar~~ manejar los Clientes disponibles en espera. Retornar el lugar a cero.

$\text{Clientes}(\text{Inaugurar}(c, p)) \equiv c$ ✓

$\text{Clientes}(\text{Alquiler}(v, c, p)) \equiv \text{Clientes}(v)$ ✓

// las restricciones de Alquiler(...) me garantizan dato existente.

$\text{Clientes}(\text{devolver}(v, c, p)) \equiv \text{Clientes}(v)$ ✓

// devolver simplifica la existencia del Cliente y por lo tanto tiempos vencidos u

// otros problemas. Dado por devolver(...) simplifica ser Cliente de existir tiempos vencidos

// le brinda automaticamente ya no sirve Cliente y no podrá devolver.

Seguimos ↓

$Clientes(Pasado(v)) \equiv ClientesPorDejar(v, Clientes(v))$

$ClientesPorDejar : VC^V \times Conj(Clubes)^C \rightarrow Conj(Clubete) \quad \{c \in clientes(v)\}$

$ClientesPorDejar(v, Clientes) \equiv$

if ($\emptyset?(Clientes)$)

then: \emptyset

else: if ($tienevalor(diasdestibolpular(v, dameUno(Clientes)), 1)$)

then: $ClientesPorDejar(v, sinUno(Clientes))$

else: $\{dameUno(Clientes)\} \cup ClientesPorDejar(v, sinUno(Clientes))$

fi ✓

// La idea de la función es verificar cada uno de todos los clientes este a 1 día de
// expirar. entonces el $Pasado(-)$ ese cliente debe ser eliminado de los
// Clientes del VideoClub. La meta última es la reconstrucción del conjunto
// Clubete a Clubete, sabiendo a fin debe eliminarse.

EJERCICIO 3

- (4) El contexto del enunciado nos ~~define~~ ^{define} que los participantes de este juego son los caballos, y se está en un establo. Si bien es útil el modo de poder visualizar, lo más relevante es entender que son participantes de una competencia, bajo un contexto y reglas de juego, comportamiento y objetivo.

De esa way, los ítems relevantes a modelar son:

- Al empezar es fijo la cantidad de participantes ✓
- El comportamiento de los animales.
en estos casos los participantes se pueden o no relacionar. esa relación es Seguidor-Líder donde un líder puede tener muchos seguidores y un seguidor solo un líder.
es importante también que esta relación es dinámica. con un participante puede elegir cambiar de líder, pero también puede ser forzado a ~~dejar~~ (cuando los seguidores deben pasar de un caballo a otro) ✓
- Los participantes se pueden eliminar entre ellos (maneuver de finta).
Además esto trae la consecuencia de modificar el líder de todos aquellos que acaban de perderlo. Más precisamente el participante que eliminó al antiguo líder. ✓

Los aspectos relevantes para diferenciarlos son:

- Los participantes ✓
- Su relación Líder-Seguidor ✓
- Desde la relación "sigue a otro y directo" entiendo que el histórico lo es relevante y los juegos son jugados en un momento de la si respetar lo antes dicho. ✓

genero: JuegoCalamar.

(b) ~~genero~~ Calamar es STRING.

$doCalamars: \text{JuegoCalamar} \rightarrow \text{conj}(\text{Calamar})$

// este observador permite distinguir que los Calamars en juego son los mismos. Retorne el conjunto de Participantes. ✓

$doSeguidores: \text{JuegoCalamar} \times \text{Calamar} \rightarrow \text{conj}(\text{Calamar}) \{c \in doCalamars(jc)\}$

// este observador permite distinguir dos Participantes teniendo en cuenta sus

// Seguidores. es decir dos Calamars son iguales no solo por un mismo

// Identificador (nombre, id, etc) sino tambien si lo siguen los mismos

// Calamars dentro del juego. ✓

igualdad Observacional:

$(\forall j_1, j_2: \text{JuegoCalamar})$ (

$j_1 \approx_{OBS} j_2 \Rightarrow$ (

$doCalamars(j_1) = doCalamars(j_2) \wedge_L$

$(\forall c: \text{Calamar}) (c \in doCalamars(j_1) \Rightarrow_L$

$doSeguidores(j_1, c) \approx_{OBS} doSeguidores(j_2, c)) \checkmark$

Dos juegos son iguales observacionalmente cuando tienen los mismos Participantes y la misma relación líder-seguidor. ✓

generadores

(c) $\text{newJgo} : \text{conj}(\text{Colabor}) \rightarrow \text{JgoColabor} \checkmark$

// este generador de inicio es la Competencia. Modela fe a
// iniciar esto en conjunto de Colaboras fe son los participantes Asle G3.

$\text{Seguir} : \text{Colabor } c_1 \times \text{Colabor } c_2 \times \text{JgoColabor} \rightarrow \text{JgoColabor}$
 // este generador permite fe c_1 comienza a seguir a c_2 $\{c_1, c_2 \in \text{derColaboras}(j_c)\}$

$\text{Menciar} : \text{Colabor } c_1 \times \text{Colabor } c_2 \times \text{JgoColabor} \rightarrow \text{JgoColabor}$
 // este generador permite modelar fe c_1 elimina a c_2 mencionando
 // con tinta \checkmark

(d) Axiomatizar:

$\text{derColaboras}(\text{newJgo}(\text{conj}(\text{Colabor } c))) \equiv c. \checkmark$

$\text{derColaboras}(\text{Seguir}(c_1, c_2, j_c)) \equiv \text{derColaboras}(j_c) \checkmark$

$\text{derColaboras}(\text{Menciar}(c_1, c_2, j_c)) \equiv \text{derColaboras}(j_c) - \{c_2\} \checkmark$

$\text{derSeguidos}(\text{newJgo}(c)) \equiv \emptyset \checkmark$

// Los seguidores de cualquier participante son el conjunto vacío. el iniciar el Jgo.

$\text{derSeguidos}(\text{Seguir}(c_1, c_2, j_c)) \equiv \exists f (c = c_2)$

+ tienen derSeguidos(j_c, c) $\cup \{c_1\}$
 + else: derSeguidos(j_c, c) \checkmark

fe

faltaba caso en que c_1 siga a c_2
 cambio de líder implique recordo
 de los seguidores de c .

estos 3 no tienen
 relación con seguimiento
 entre los colaboradores, lo
 axiomatiza igual por
 ser.
 No lo hago por feoligidad.
 el axioma fe infiere la
 relación entre colaboradores
 es derSeguidos(\dots).
 OK!

5.70 de 100

derSeguidores (Menores $(c_1, c_2, jc), c$) \equiv

IF $(c = c_1)$

+ Tnen derSeguidores(c) U derSeguidores(c₂)

+ else: derSeguidores(c) - {c₂}

fi

// dada la respuesta en derSeguidores() $c = c_2$ debe de ser un
// participante y entonces no debe ser excluido. ✓

Aclaración:

- Como pedía el enunciado definir observadores y generadores, además de la igualdad dimensional, y así como fue son suficientes para reflejar la información solicitada.

Con respecto a la aclaración del enunciado "Saber en todo momento quienes van ganando el juego" sería necesario construir otras operaciones para resolverlo.

Un ejemplo podría ser:

derSeguidores: $jc \rightarrow \text{Conj}(\text{Calenmz})$

donde se realiza la especificación por ~~parte~~ retorne en el conjunto de
Saber solo ~~los~~ los del Calenmz con más seguidores. OK! 😊